UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Optimal priority ordering in optimal PHP production of multiple part-types on a failure-prone machine with quadratic buffer cost

Albert Corominas, Robert Griñó, Rafael Pastor

IOC-DT-P-2004-19
Octubre 2004

Institut d'Organització i Control
de Sistemes Industrials

Optimal Priority Ordering in Optimal PHP Production of Multiple Part-Types on a Failure-Prone Machine with Quadratic Buffer Costs

Albert Corominas, Robert Griñó, *Member, IEEE*, Rafael Pastor

This work was supported in part by the *Comisión Interministerial de Ciencia y Tecnologia (CICYT)* under project DPI2000-1509-C03-02,03.

A. Corominas, R. Griñó and R. Pastor are with the *Instituto de Organización y Control de Sistemas Industriales*, *Universitat Politècnica de Catalunya*, Barcelona, Spain. E-mail: {albert.corominas, roberto.grino, rafael.pastor}@upc.es

Abstract

Shu and Perkins [1] deals with the problem of minimising the expected sum of quadratic buffer costs when a single, failure-prone machine produces multiple part-types. They restrict the set of control policies to the class of prioritised hedging point (PHP) policies and determine simple, analytical expressions for the optimal hedging points, provided that the priority ordering of the part-types is given. This paper addresses the determination of the optimal priority ordering for PHP policies and reports the results of a computational experiment. The conclusions are that instances of up to approximately twenty-five part-types can be solved to optimality in short computing times, that it is worthwhile to use dominance relations and that the influence of the values of some parameters is insignificant.

Index Terms

Prioritised hedging point control, scheduling policies, production control, manufacturing systems.

I. Introduction and Problem Statement

Shu and Perkins [1] deals with optimising the costs of a special production system; namely, a single, failure-prone machine that is able to simultaneously produce up to n part-types. The demand arrival rates for the part-types ($d_j;\ j = 1,\ 2, ...,\ n$) are assumed to be constant. The machine alternately adopts an "up" state in which it is fully functional or a "down" state in which it is not able to produce anything. The time that the machine spends in each state, before switching to the other, is exponentially distributed with averages equal to $1/q_d$ and $1/q_u$ for the "down" and "up" states respectively. The system control is $u(t) = [u_1(t),\ u_2(t), ..., u_n(t)]$, which is the vector of the rates of production for each part-type. The buffer level of part-type j at time t, $x_j(t)$ satisfies the equation $\dot{x}_j(t) = u_j(t) - d_j$. Without loss of generality, it is assumed that the maximum production rate for any part-type is equal toμ, which is the capacity of the machine; therefore, the production rates, $\forall t \geq 0$, must fulfil the condition$\sum_{j=1}^{n} u_j(t) \leq \mu$. It is also assumed that the condition$\frac{q_u}{q_d+q_u} \cdot \mu - \sum_{j=1}^{n} d_j > 0$ is fulfilled, i.e., that the machine has enough capacity to meet demand. The instantaneous cost function of the system is assumed to be equal to$\sum_{j=1}^{n} c_j \cdot x_j^2(t)$, where c_j ($j = 1,\ 2, ...,\ n$) are nonnegative constants. The objective, then, is to minimise the expected long-term average cost:

$$J = \lim_{T\to\infty} \frac{1}{T} E\left[\int_0^T \left(\sum_{j=1}^{n} c_j \cdot x_j^2(t)\right) dt\right]$$

The problem is presented in [1], as an extension of previous works [2], [3], [4]. The optimal general solution of the problem is not known; however, simple closed-form expressions are established in [1] for the optimal hedging points of the part-types, which take into account the control policies in the class of prioritised hedging point policies. In this class of policies a priority ordering, $\{p_1, p_2, ..., p_n\}$, is established for the part-types and the machine attempts to drive the buffer level of each part-type to its hedging point, z_{p_j}, and to keep it at this level. When the machine is "up", its production capacity is assigned as follows (when it is "down", the machine cannot work at all). As for part-typep_1, if the difference $z_{p_1} - x_{p_1}(t)$ is positive, zero or negative (this can only happen in a transient state), the rate of production $u_{p_1}(t)$ is respectively equal toμ, d_{p_1} or 0. As regards any other part-type p_j ($j > 1$), its rate of production is equal to zero unless the buffer levels of all the

part-types with greater priority are above or equal to their corresponding hedging points; when this condition is fulfilled, if the difference $z_{p_j} - x_{p_j}(t)$ is positive, zero or negative (this also can only happen in a transient state), the rate of production $u_{p_j}(t)$ is respectively equal to μ minus the capacity assigned to those part-types with greater priority $\left(\mu - \sum_{s=1}^{j-1} d_{p_s}\right)$, d_{p_j} or 0.

In [1], expressions are provided for calculating the optimal cost corresponding to a given priority ordering of the part-types. However, only partial results concerning the optimal priority ordering are included in the paper and its authors point out that this remains an area for future research.

II. Determination of optimal priority orderings

In [1] it is shown that the expected cost corresponding to a part-type does not depend on the order of the part-types with greater priority, but only on the sum of all their demand rates. It is also shown that when a part-type i *dominates* a part-type j, placing i directly before j yields a cost that is not greater than the one corresponding to placing j directly before i(i.e., in order to find an optimal priority ordering there is no need to take into account orderings in which j is placed immediately before i). In this paper, we adopt a definition of the dominance relation that differs slightly from the one proposed in [1], which thus avoids the possibility of mutual dominance between a pair of part-types and, at the same time, preserves the aforementioned property; we can say that i *dominates* j if and only if one of the three following conditions is fulfilled: (i) $c_i > c_j$ and $c_i d_i \geq c_j d_j$; (ii) $c_i = c_j$ and $c_i d_i > c_j d_j$ or (iii) $c_i = c_j$, $d_i = d_j$ and $i < j$.

These two properties allow one to reduce the enumerative effort involved in finding an optimal priority ordering. If there are no domination relations between pairs of part-types, or if one does not use these relations, the computational complexity of the calculations is proportional to $\sum_{k=1}^{n} n \cdot \begin{pmatrix} n-1 \\ k-1 \end{pmatrix} = n \cdot 2^{n-1}$. Of course, the domination relations, insofar as they reduce the number of orderings to be taken into account, can contribute towards reducing the computational effort.

Taking into account these properties, the determination of an optimal priority ordering can be thought of as a deterministic dynamic programming problem, in which the stages correspond to positions in the priority ordering and states are characterised by the sets of already prioritised part-types. If dominance relations are used, the characterisation of a state must also include an indication of whatever is the last element in the set. The decision to be made, at each state, consists in determining the next part-type in the priority ordering; this part-type must be one of those not already prioritised and, if dominance relations are used, one not dominated by the last part-type in the partial preceding priority ordering.

Of course, the limitations of this procedure stem from the fact that the number of states increases exponentially as n increases$n$. It is straightforward that for $n > 1$ part-types the maximum number of states$n > 1$ is reached at stage $k = \left\lceil \frac{n+1}{2} \right\rceil$ (and also, when $\frac{n+1}{2}$ is an integer, at $k = \left\lceil \frac{n+1}{2} \right\rceil - 1$). For instance, the number of states for $n = 23$ at stages 11 and 12 is equal to 1,352,078).

III. Computational experiment

Two dynamic programming codes were used in the computational experiment: one that took into account dominance relations, and one that did not. We will denominate these two codes DP-Y and DP-N respectively.

The latter is simpler than the former, however as DP-N has to take into account a greater number of states, it was not possible to determine a priori which would be the most efficient.

Both codes were applied to the same set of instances. The set consisted of 1,400 instances (100 instances for each value of n in $[10, 23]$); the values of d_j and c_j were generated at random using uniform discrete distributions in $[1, 100]$ and $[1, 20]$, respectively.

In order to guarantee that the machine had enough capacity to meet the demand, the value of μ was set equal to $\alpha \cdot \frac{q_d+q_u}{q_u} \cdot \sum_{j=1}^{n} d_j$, where $\alpha > 1$.

The experiment was performed on a PC Pentium IV, at 1.8 GHz , with 512 Mb RAM.

In order to evaluate the influence of the values of α and of the ratio $\frac{q_d+q_u}{q_u}$ on the computing times, another experiment was performed prior to this using both codes, DP-Y and DP-N. It used 25 instances in which $n = 18$ and the values $\alpha = 1.05, 1.10, 1.25, 1.50, 2.00$ and $\frac{q_d+q_u}{q_u} = 3.00, 2.00, 1.50, 1.20, 1.10$. For the different combinations of the values of the parameters, the computing times to solve each instance turned out to be almost identical. We concluded that the influence of the parameters α and $\frac{q_d+q_u}{q_u}$ on the computing time was not significant and therefore their values could be fixed (specifically, we established that $\alpha = 1.10$ and $\frac{q_d+q_u}{q_u} = 1.20$, and therefore that $\mu = 1.32 \cdot \sum_{j=1}^{n} d_j$).

As concerns the continuation of the experiment, Table I shows the minimum, average and maximum computing times, in seconds, that correspond to the application of code DP-N (i.e., without taking into account dominance relations).

n	Min.	Av.	Max.
10	0	0.08	1
11	0	0.18	1
12	0	0.41	1
13	0	0.90	1
14	1	2.05	3
15	4	4.58	5
16	9	10.09	11
17	21	22.30	23
18	47	48.83	51
19	102	106	112
20	222	231	248
21	487	510	533
22	1046	1092	1147
23	2233	2358	2541

TABLE I

MINIMUM, AVERAGE AND MAXIMUM COMPUTING TIMES (IN SECONDS) WITH DP-N.

One can appreciate the little dispersion of the computing times for a given value of n and furthermore that the ratio between the average computing times corresponding to n and to $n-1$ is approximately equal to $2 \cdot \frac{n}{n-1}$, as could be expected from an analysis of the complexity of the algorithm in [1]. Fig. 1 shows the

average computing time as a function of the number of part-types, n.

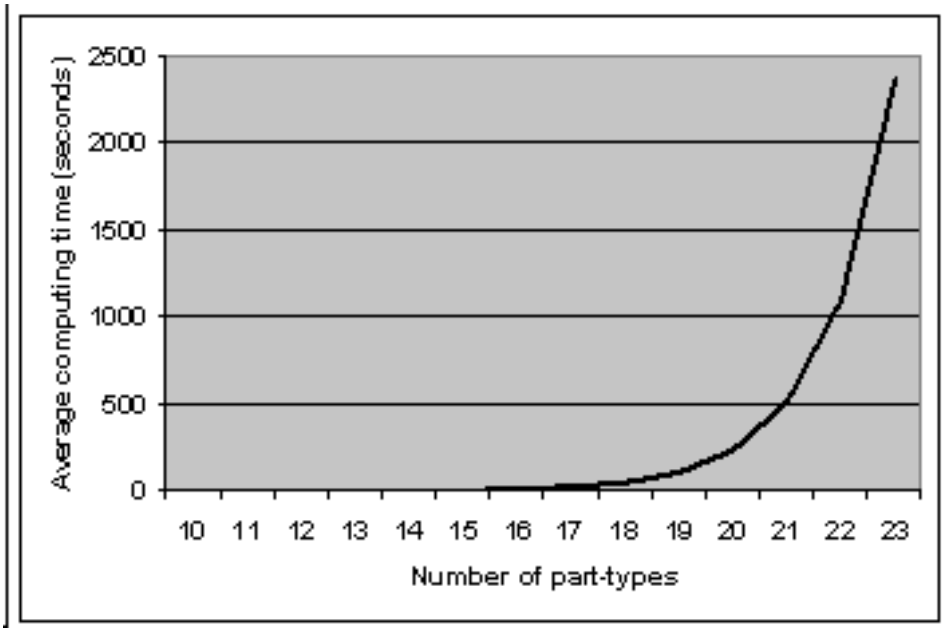

Fig. 1. Average computing time for DP-N versus number of part-types.

The computing times using code DP-Y, which does take into account dominance relations, depend on the density of these relations, which is defined as the ratio between the number of actual dominance relations corresponding to the instance and its maximum possible value (which is equal to $n \cdot (n-1)/2$). Table II contains the minimum, average and maximum density of dominance relations corresponding to the set of instances used in the experiment, expressed as percentages, and Table III the minimum, average and maximum computing times, in seconds, using DP-Y.

In overall terms, when dominance relations are used the computing times are shorter than those obtained when they are not used. The dispersion for a given value of n is, however, greater, as was expected. The average computing times increase exponentially with the value of n, as occurs when dominances are not taken into account.

To analyse the influence of the number of dominance relations on the computing time in greater depth, in Fig. 2 one can see, for the instances for which $n = 23$, the computing times that correspond to solving the instances versus the number of dominance relations. The figure shows a decreasing trend in the computing time, as the number of dominances increases, with a correlation coefficient equal to -0.578.

IV. Conclusions and research prospects

The computational experiment to determine optimal priority orderings for prioritised hedging point control policies shows that using dominance relations is worthwhile and that instances with up to approximately 25 part-types can be solved in relatively short computing times. However, given the exponential increases, as n increases, of the required memory and the computing time, such a solution, taking advantage of the procedures

n	Min.	Av.	Max.
10	35.6	76.2	97.8
11	47.3	76.0	98.2
12	48.5	78.4	97.0
13	57.7	75.8	92.3
14	56.0	75.6	93.4
15	56.2	77.8	95.2
16	50.0	77.2	94.2
17	54.4	76.5	91.2
18	57.5	75.5	88.9
19	59.1	75.9	90.6
20	59.5	76.7	88.4
21	58.1	77.6	94.8
22	61.0	75.9	87.0
23	50.2	75.5	88.9

TABLE II

MINIMUM, AVERAGE AND MAXIMUM DENSITY OF DOMINANCE RELATIONS (IN %).

n	Min.	Av.	Max
10	0	0.03	1
11	0	0.09	1
12	0	0.17	1
13	0	0.40	1
14	0	0.86	2
15	1	1.79	3
16	2	4.11	7
17	5	8.53	19
18	10	18.80	39
19	21	39.30	78
20	45	80.12	144
21	96	179	395
22	194	382	854
23	439	769	1518

TABLE III

MINIMUM, AVERAGE AND MAXIMUM COMPUTING TIMES (IN SECONDS) USING DP-Y.

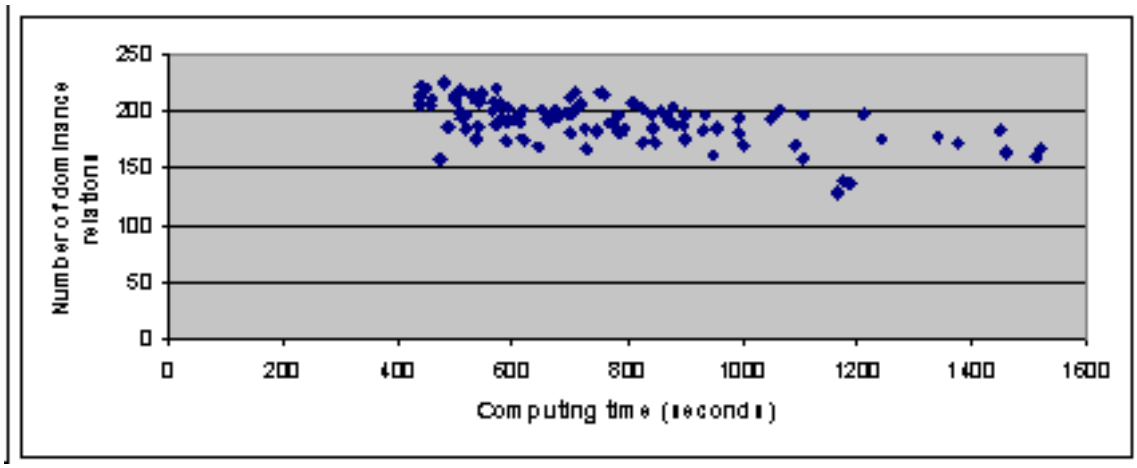

Fig. 2. Computing time versus number of dominance relations for $n = 23$.

used in the computational experiment, is prohibitive for large values of nn. This is due that fact that even if one uses more powerful hardware and allows for more computing time, the number of part-types cannot be greater than some tens. In order to solve larger instances, the authors are investigating the use of heuristics and the introduction of bounds in a dynamic programming scheme.

REFERENCES

[1] C. Shu and J. Perkins, "Optimal PHP production of multiple part-types on a failure-prone machine with quadratic buffer costs," vol. 46, pp. 541–549, Apr. 2001.

[2] J. Perkins and R. Srikant, "Scheduling multiple part-types in a unreliable single-machine manufacturing system," vol. 42, pp. 364–377, Mar. 1997.

[3] ——, "Hedging policies for failure-prone manufacturing systems: Optimality of JIT and bounds on buffer levels," vol. 43, pp. 953–957, July 1998.

[4] ——, "Failure-prone production systems with uncertain demand," vol. 46, pp. 441–449, Mar. 2001.